Eurographics Symposium on Geometry Processing 2013
Yaron Lipman and Richard Hao Zhang
(Guest Editors)

Volume 32 (2013), Number 5

Watertight Scenes from Urban LiDAR and Planar Surfaces

M. van Kreveld[1], T. van Lankveld[2], and R. C. Veltkamp[1]

[1]Department of Information and Computing Sciences, Utrecht University, The Netherlands
[2]Geometrica, Inria - Sophia Antipolis, France

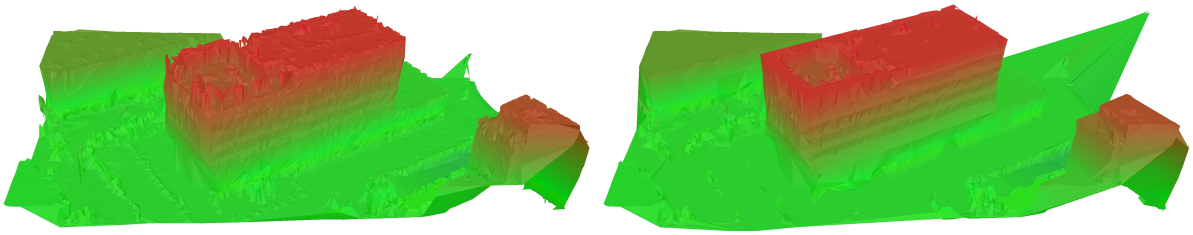

Figure 1: *Watertight geometry from points (left) and surfaces (right)*

Abstract

The demand for large geometric models is increasing, especially of urban environments. This has resulted in production of massive point cloud data from images or LiDAR. Visualization and further processing generally require a detailed, yet concise representation of the scene's surfaces. Related work generally either approximates the data with the risk of over-smoothing, or interpolates the data with excessive detail. Many surfaces in urban scenes can be modeled more concisely by planar approximations. We present a method that combines these polygons into a watertight model. The polygon-based shape is closed with free-form meshes based on visibility information. To achieve this, we divide 3-space into inside and outside volumes by combining a constrained Delaunay tetrahedralization with a graph-cut. We compare our method with related work on several large urban LiDAR data sets. We construct similar shapes with a third fewer triangles to model the scenes. Additionally, our results are more visually pleasing and closer to a human modeler's description of urban scenes using simple boxes.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Geometric algorithms; Surface and solid representations; Computer Graphics [I.3.7]: Three-dimensional Graphics and Realism—Visible line/surface algorithms

1. Introduction

Public interest in virtual cityscapes has been increasing for some years. Groups like municipal government, architecture firms, and emergency response units can use these models for urban planning, visualization, and training. As a result, image-based point reconstruction and LiDAR sensing have received a wide backing, yielding massive data sets of high detail point clouds. However, these point data are not sufficient for most applications. Consequently, reconstructing surfaces from point data is an active field of research.

Many applications prefer the geometry to be non-self-intersecting, while applications like disaster management simulations may require the surfaces to be enclosing a volume as well, i.e. watertight. Watertight geometry divides 3-space into two distinct volumes, inside and outside the model. The surfaces of the model are exactly where these two volumes meet.

Reconstructing watertight geometry has received considerable attention, as shown in Section 2. However, most related methods do not incorporate the fact that most surfaces in urban scenes are planar. This generally results in rounding of sharp corners or adding unnecessary complexity to the model. In Section 3, we present a novel method that con-

structs a concise watertight model using planar polygons approximating the local point cloud. We have evaluated our method on a number of urban LiDAR data sets and we describe these experiments in Sections 4 and 5. Finally, Section 6 presents a discussion of the broader implications of our method. Our key contributions are:

- A novel method that constructs watertight geometry from points and a soup of polygons. This geometry follows the polygons where possible and closes the object based on a line-of-sight based space carving.
- A novel method for constructing the constrained Delaunay tetrahedralization that avoids costly line-sphere intersection computations.
- A comparison of our method to the related method by Labatut *et al.* [LPK09]. Our method produces concise and visually pleasing results using a third fewer triangles.

2. Related Work

For the purpose of constructing watertight geometry, implicit surface methods, e.g. [CBC*01], may be the most straightforward, because the implicit function already represents some notion of an inner and outer volume. Mullen *et al.* [MDGD*10] describe a method that improves the correctness of the local surface orientation in order to fill gaps in regions of missing data. Schnabel *et al.* [SDK09] use implicit surfaces and combine this with graph-cuts to get a watertight model. They use an optimization scheme that iterates over multiple graph-cuts. Shalom *et al.* [SSZCO10] explicitly indicate exterior regions in the implicit surface by constructing generalized cones with a data point at their apex. Shen *et al.* [SOS04] present a method for constructing implicit surfaces from a polygon soup. Alliez *et al.* [ACSTD07] create an implicit surface using the anisotropy in Voronoi cells to estimate the normals of the surface.

Implicit surface methods have two inherent weaknesses: a) they approximate, rather than interpolate the point set and because of this b) they have problems reconstructing sharp features. Many implicit surface methods are also ill equipped to handle massive data sets.

There is also a wide range of methods that construct watertight geometry using the facets of the Delaunay tetrahedralization (DT). A survey of these Delaunay-based methods is given by Cazals and Giessen [CG06]. The basis for these methods is an observation by Boissonnat [Boi84], which showed that a shape can be captured in the DT of the sample points. Dey and Goswami [DG03] provide a recent example by adjusting the Cocone method [ACDL00] to produce watertight models. Many of these Delaunay-based methods assume the object is *r*-sampled, e.g. [AAK*09], which requires the sampling to be sufficiently dense near important features. The assumption of *r*-sampling is too strong for urban reconstruction from LiDAR, because a data set may combine thin features and unpredictable sampling densities.

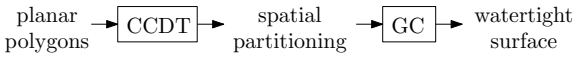

Figure 2: *An overview of our method.*

A recent approach by Labatut *et al.* [LPK09] combines the DT, the minimal-weight graph-cut, and lines-of-sight in order to determine the regions inside and outside the objects. While this method produces nice results, it does not exploit the planarity in the scene. Chauve *et al.* [CLP10] combine a partitioning of space into regions using planar primitives and a graph-cut to select interior and exterior regions. Although they achieve nice results, their method cannot reconstruct non-planar parts of a scene and lacks a theoretical basis.

Similar to [CLP10, LPK09], our method partitions space and constructs watertight geometry by applying a graph-cut to this partitioning. We partition 3-space using an extension of the DT that can handle polygons, called the *conforming constrained Delaunay tetrahedralization* (CCDT) [She98, She02, SG05]. The CCDT is constructed from a collection of points and planar polygons: each point is contained as a vertex and each polygon is contained as a collection of triangular facets. This structure is described in Subsection 3.1.

There are few methods that compute planar polygons approximating a point cloud. While various methods exist to estimate planar or near-planar approximations of a subset of a point cloud [TkG07, SWK07, TTC07], they generally do not bound these approximations based on the point cloud. Recent work by Van Kreveld *et al.* [vKvLV11] uses Efficient RANSAC [SWK07] to estimate planar approximations and an adaptation of the α-shape [EKS83] to bound the shape such that sharp features are preserved. We use these polygons as input to our method.

3. Method

The goal of our method is to construct a watertight geometric model from a collection of planar polygons. This model should contain parts of these polygons where this is useful for separating the inner and outer volumes. In regions without polygons, an appropriate boundary is estimated from the point set. Our method achieves this in two steps, as shown in Figure 2, similar to Labatut *et al.* [LPK09]. First we partition the complete space into many small regions and then we determine for each region whether it is inside or outside the model. Unlike [LPK09], both of these steps are guided by the polygons.

We use an extension of the DT, called the conforming constrained Delaunay tetrahedralization (CCDT) to partition 3-dimensional space into tetrahedral cells. We describe the CCDT and how we construct it in Subsection 3.1. If all the cells are assigned to either the inner or outer volume, the facets separating inner and outer cells comprise a watertight

model. The CCDT embeds input polygons in its facets, enabling partitioning along these surfaces.

We classify the cells of the CCDT into inner and outer cells using a minimum-weight graph-cut. This method takes a weighted graph that contains a source and a sink node and collects a number of edges to remove, the *cut*, such that the source and sink are no longer connected by a path on edges of the graph. The cut is chosen such that the sum of the weights of its edges is minimized. Subsection 3.2 describes the graph-cut method in more detail.

We construct the graph to cut from the dual of the CCDT, which is similar to a Voronoi diagram. The graph-cut will correspond exactly to the facets of the CCDT separating outer and inner cells. This collection of facets constitutes a watertight model. It may be obvious that the shape of the model reconstructed by this method depends on the shape and location of the surfaces embedded in the CCDT. If a surface is not present in the CCDT, it cannot be selected by the graph-cut. Therefore, the geometry of the CCDT has a major influence on the geometry of the reconstruction.

However, there are two other considerations important to the cut. Firstly, graph-cuts only make sense if a source and sink node exist. The connectivity between source and sink node is very important, because only edges on a path between source and sink are cut. We add two new, abstract, nodes to the dual of the CCDT to act as source and sink. These two nodes are connected automatically to the other nodes in a careful fashion as described in Subsection 3.2.1.

Secondly, the weights of the edges in the graph determine the optimal positions to cut. We compute the weight of each edge based on several properties of its corresponding facet in the CCDT. The first two factors are described in Subsection 3.2.1 and the last two in Subsection 3.2.2.

3.1. Conforming Constrained Delaunay Tetrahedralization

We subdivide 3-space into regions that can be inside or outside the model using the CCDT. This structure is closely related to the well-known DT. In order to properly describe the CCDT, we must first describe the piecewise linear complex (PLC) [MTT*96]. A PLC is a collection of points, straight line-segments, and planar straight-line polygons in 3-space. The polygons can be non-convex with any number of boundary segments and they may contain holes, interior segments, and isolated points. However, their boundary cannot be self-intersecting and all their vertices must be coplanar.

As its name implies, a PLC is a complex: the collection is both closed and non-intersecting. By closed we mean that the collection contains all faces of each of its elements. In other words, if a PLC contains a polygon, then it must also contain all the edges and vertices of the polygon. By non-intersecting we mean that the collection does not contain two

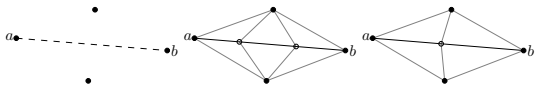

Figure 3: *A configuration of points and two different configurations of Steiner points that force $\overline{ab}$ to appear in the Delaunay triangulation.*

elements that pairwise intersect in their interiors. That is, no two elements, whether polygon, segment, or point, may intersect except in the union of their shared vertices and edges.

The CCDT is based on the *constrained Delaunay tetrahedralization* (CDT). Some of the facets of this tetrahedralization are marked as *constrained* and the tetrahedrons are Delaunay with the exception that their circumscribing ball may contain points on the other side of a constrained facet. A CDT *embeds* a PLC if it contains all the points and segments of the PLC and each polygon of the PLC is exactly covered by a collection of constrained facets.

Shewchuk [She98] showed that there are PLCs that do not admit an embedding CDT. He goes on to prove that for any PLC X we can construct another PLC $\bar{X}$ that can be embedded in a CDT. This PLC $\bar{X}$ exactly covers X, but it contains additional points, called *Steiner points*, on the segments of X. These points are placed such that the DT of $\bar{X}$ contains edges that cover the segments of X. We call these edges *conforming* to the segments of X. Unlike Shewchuk, we prefer to call the CDT of $\bar{X}$ the conforming constrained Delaunay tetrahedralization of X, to indicate the similarity to the 2-dimensional conforming Delaunay triangulation.

3.1.1. Embedding Segments

There are many different methods for determining the positions of the Steiner points that ensure that the segments of $\bar{X}$ are present in the DT. Figure 3 gives an example PLC and two different configurations of Steiner points that ensure the embedding of the segment $\overline{ab}$.

There are also various ways of comparing the quality of configurations, e.g. by counting the number of Steiner points [SG05] or by bounding the minimal edge length [She02]. Related work generally constructs Steiner points at the intersections of a sphere and a line, which is in practice either dangerous because of round-off errors or expensive to compute exactly. For this reason, we present a novel Steiner point insertion method, similar to [SG05], that does not require computing sphere-line intersections.

A recurring concept in methods for constructing conforming edges is the *protecting ball* of a point. The purpose of a protecting ball is to indicate a region around an input point where no Steiner point may be placed. Let us consider the balls $\mathcal{B}_p^q$ centered on p and touching a vertex or edge q of X in their boundary. The ball $\mathcal{B}_p$ is the smallest of these balls

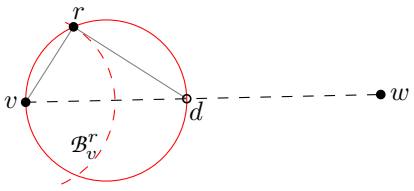

Figure 4: *A protecting ball. The black disks are points of the PLC, the dashed black line is a segment of the PLC. The dashed red arc shows $\mathcal{B}_v^r$, the solid red circle is the largest empty ball with its diameter $\overline{vd}$ on $\overline{vw}$.*

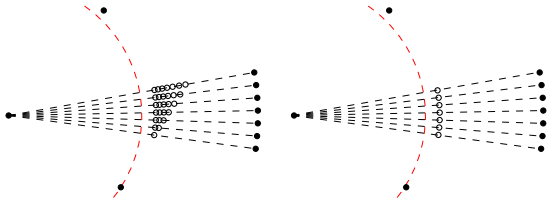

Figure 5: *A pathological case when embedding conforming segments top to bottom between points (black disks). Left: inserting Steiner points (circles) at d; each Steiner point insertion invalidates all earlier conforming edges, leading to repeated Steiner point insertions. Right: inserting Steiner points at m, as shown in Figure 6.*

$\mathcal{B}_p^q$. The protecting ball $\mathcal{P}_p$ of a point p is an open ball centered at p no larger than $\mathcal{B}_p$. The radius of each $\mathcal{P}_p$ is generally chosen based on the local distribution of points and conforming edges. Let us assume for now that $\mathcal{P}_p = \mathcal{B}_p$.

These protecting balls are specifically important for *acute* points: points where at least two conforming edges meet at an acute angle. The insertion of a Steiner point on one edge may cause another conforming edge e to disappear from the DT, forcing the insertion of another Steiner point to restore e. When Steiner points can be inserted arbitrarily close to an acute point, this process may cascade into an infinite loop.

In order to embed a conforming segment $\overline{vw}$ not present in the DT, we need to insert Steiner points. Let us consider the Steiner point s whose insertion results in the appearance of $\overline{vs}$ in the DT. This Steiner point may not be placed inside $\mathcal{P}_v$. However, in order for $\overline{vs}$ to appear in the DT, there must be a sphere through v and s with interior void of points. Most related methods will place s on the boundary of $\mathcal{P}_v$. Because $\mathcal{P}_v$ cannot contain any point, the open ball $\mathcal{D}_{\overline{vs}}$ with diameter $\overline{vs}$ must be empty. Therefore s meets both criteria: it is outside $\mathcal{P}_v$ and shares the boundary of an empty ball with v.

Instead of computing the intersection of the boundary of $\mathcal{P}_v$ and $\overline{vw}$, we will work directly with the empty open balls incident to v that have their diameter on $\overline{vw}$, as shown in Figure 4. Specifically, we choose the largest empty open ball $\mathcal{D}_{\overline{vd}}$. This ball must have a point r on its boundary and it is easy to see that $\|v,r\| < \|v,d\|$. Note that r may be a point of X or a Steiner point inserted earlier.

Because d is not inside $\mathcal{B}_v^r$ and because all Steiner points are constructed on the segments of X, d must be outside $\mathcal{P}_v$. Because $\mathcal{D}_{\overline{vd}}$ is empty, inserting d as Steiner point would result in the $\overline{vd}$ appearing in the DT. Additionally, v and r must be adjacent in the DT as witnessed by $\mathcal{D}_{\overline{vd}}$. Finally, according to Thales' Theorem we can compute d as the intersection of $\overline{vw}$ and the plane through r orthogonal to $\overline{vr}$, removing the need for a line-sphere intersection.

Unfortunately, inserting a Steiner point at d has two disadvantages. Firstly, if we want to be able to insert conforming edges incrementally, this method may result in an extreme number of Steiner points, as shown in Figure 5 (left). Secondly, if there are two points x,y cospherical with v,r,d, $\overline{vd}$ may not appear in the DT because the tetrahedra on v,r,x,y and d,r,x,y have an empty circumsphere. The free choice of which tetrahedra to use may not result in $\overline{vd}$ appearing.

Algorithm 1 Compute the position of a new Steiner point

Input: two vertices v,w.
Output: the Steiner point inserted on $\overline{vw}$ to protect v

```
1:  procedure PROTECT(v, w)
2:      x ← w
3:      y ← w
4:      E ← edges incident to v
5:      for all v̄v̄_i ∈ E do
6:          if ∠wvv_i < π/2 then
7:              P ← plane through v_i orthogonal to v̄v̄_i
8:              d ← v̄w̄ ∩ P
9:              if ||v,d|| < ||v,y|| then
10:                 x ← projection of v_i on v̄w̄
11:                 y ← d
12:             end if
13:         end if
14:     end for
15:     return MIDPOINT(x, y)
16: end procedure
```

In order to overcome these disadvantages, we want to insert the Steiner point at another location s on $\overline{vw}$ such that $\|v,r\| \leq \|v,s\| < \|v,d\|$. We choose as s the midpoint m of d and the projection p of r onto $\overline{vw}$, as shown in Figure 6 and Algorithm 1. We refer to this procedure as the *protection method*.

It is straightforward to see that $\|v,p\| < \|v,m\| < \|v,d\|$ and $\|v,p\| < \|v,r\|$. However, the difference between $\|r,v\|$ and $\|m,v\|$ depends on the angle $\theta = \angle rvw$. The following lemma shows that for any θ, $\|v,r\| < \|v,m\|$ and therefore m must lie strictly outside the protecting ball and strictly inside $\mathcal{D}_{\overline{vd}}$. This means that $\overline{vm}$ must appear in the DT.

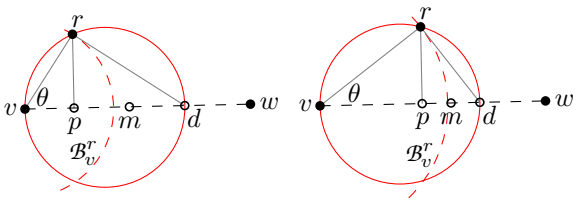

Figure 6: *Two protecting balls. The black disks are vertices of the PLC, the dashed black line is a segment of the PLC, and the black circles indicate other significant locations. The dashed red arc shows $\mathcal{B}_v^r$, the solid red circle is the largest empty ball with its diameter $\overline{vd}$ on $\overline{vw}$, p is the projection of r onto $\overline{vw}$, and m is the mid-point of p and d.*

Lemma 1 *Given the triangle vrd where r lies on a corner with right angle, let p be the orthogonal projection of r onto $\overline{vd}$, let m be the midpoint between p and d, and let $\theta = \angle rvd$, as shown in Figure 6. If $\theta \neq 0$ then $\|v,r\| < \|v,m\|$.*

Proof For readability's sake, let us rename the distance involved as follows: $R = \|v,r\|$, $A = \|r,d\|$, $B = \|v,p\|$, $C = \|p,d\|$, $D = \|r,p\|$, as shown in Figure 7. Observe that $\angle prd = \theta$, because of triangle similarity. This proof builds on the following trigonometric functions:

$$B = R\cos\theta \quad (1)$$
$$D = R\sin\theta \quad (2)$$
$$D = A\cos\theta \quad (3)$$
$$C = A\sin\theta \quad (4)$$

We will show that the following relation between the distances $\frac{R-B}{C} < \frac{1}{2}$ holds for any $\theta \neq 0$. In order to achieve this result, we will apply the above equalities in order.

$$\begin{aligned}\frac{R-B}{C} &= \frac{R-R\cos\theta}{C} = \frac{R(1-\cos\theta)}{C} \\ &= \frac{D(1-\cos\theta)}{C\sin\theta} \\ &= \frac{A\cos\theta(1-\cos\theta)}{C\sin\theta} \\ &= \frac{A\cos\theta(1-\cos\theta)}{A\sin^2\theta} = \frac{\cos\theta-\cos^2\theta}{\sin^2\theta} \\ &< \frac{1}{2}\end{aligned}$$

The final inequality follows from evaluating this function at $\lim_{\theta\to 0}$, where it reaches its maximum because of the similarity to $\tan(\theta)^{-2}$. The function cannot be evaluated at $\theta = 0$, but in this case $r = d$ and $\overline{vr}$ already exists in the DT. □

The protection method automatically constructs a protecting region around the input vertices based on the local point distribution. This region is roughly ball-shaped with dents at nearby vertices of X. There are two remaining issues when conforming a DT to a PLC X. Firstly, the parts of the segments of X between these protecting regions also need to be embedded in the DT. Secondly, it may occur that a segment $\overline{vw}$ of X is not embedded in the DT if the balls $\mathcal{B}_v$ and $\mathcal{B}_w$ intersect. In order to conform the DT to $\overline{vw}$, a Steiner point must be constructed inside either $\mathcal{B}_v$ or $\mathcal{B}_w$.

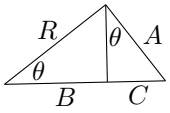

Figure 7: *The setup for the proof of Lemma 1.*

The parts of the segments of X between protecting regions are easily embedded by recursively applying the protection method for the newly inserted Steiner points. Starting from a segment $\overline{vw}$, Steiner points s_v, s_w are constructed such that the edges $\overline{vs_v}, \overline{ws_w}$ appear in the DT. Then, the segment $\overline{s_v s_w}$ is embedded similarly.

When two points v,w lie sufficiently close together and $\overline{vw}$ is not present in the DT, it may be necessary to insert a Steiner point inside $\mathcal{B}_v$ or $\mathcal{B}_w$. Related methods achieve this by choosing the protecting balls $\mathcal{P}_v$ and $\mathcal{P}_w$ smaller than $\mathcal{B}_v$ and $\mathcal{B}_w$ [She02, SG05]. Similarly, we will construct a Steiner point s such that $\overline{vs}$ and $\overline{sw}$ appear in the DT, while at the same time forcing $\mathcal{B}_v$ or $\mathcal{B}_w$ to shrink. We must choose s with caution in order to make sure no $\mathcal{B}_v$ can shrink too much.

We will distinguish three different cases based on the vertices v,w and the Steiner points s_v, s_w that would be constructed to protect them. Case i) occurs when the segments $\overline{vs_v}$ and $\overline{ws_w}$ do not overlap. In this case we compute the Steiner points for both ends using our protection method. Note that if $\mathcal{B}_v$ and $\mathcal{B}_w$ overlap, then $\overline{vs_v}$ and $\overline{ws_w}$ must overlap.

Case ii) occurs when $\overline{vs_v}$ and $\overline{ws_w}$ overlap and one of the points v,w is acute and the other is not. Note that Steiner points are never acute. Let us assume without loss of generality that the acute point is v. We insert only the Steiner point s_v into the PLC. This does not reduce the size of $\mathcal{B}_v$, and because w is not acute, none of the conforming edges incident to w can be removed by this insertion.

Case iii) covers all remaining configurations of v and w where $\overline{vs_v}$ and $\overline{ws_w}$ overlap. Again, we will insert one Steiner point s inside $\mathcal{B}_v$ or $\mathcal{B}_w$. We base s on s_v, s_w, and the midpoint m of v,w. Let us assume without loss of generality that $\|v,s_v\| \leq \|w,s_w\|$. If $\|v,s_v\| < \|v,m\|$ then $s = s_v$; otherwise $s = m$. After the insertion of s, the radius of both $\mathcal{B}_v$ and $\mathcal{B}_w$ either remains the same or is at least $\frac{\|v,w\|}{2}$. The pseudo-code of the complete method is shown in Algorithm 2.

So far we have mainly considered inserting a single conforming segment. Figure 6 (right) shows the Steiner points constructed on multiple segments.

It remains to show that our method always terminates.

Algorithm 2 Embed a conforming edge

Input: two vertices v, w.

```
1:  procedure CONFORM(v, w)
2:      if exists(vw) then
3:          return
4:      end if
5:      s_v ←PROTECT(v, w)
6:      s_w ←PROTECT(w, v)
7:      if ||v, s_v|| + ||w, s_w|| > ||v, w|| then          ▷ Case i)
8:          insert s_v
9:          insert s_w
10:         CONFORM(s_v, s_w)
11:     else if acute(v) and ¬acute(w) then                 ▷ Case ii)
12:         insert s_v
13:         CONFORM(s_v, w)
14:     else if acute(w) and ¬acute(v) then                 ▷ Case ii)
15:         insert s_w
16:         CONFORM(v, s_w)
17:     else                                                ▷ Case iii)
18:         m ←MIDPOINT(v, w)
19:         if ||v, s_v|| < ||w, s_w|| and ||v, s_v|| < ||v, m|| then
20:             insert s_v
21:             CONFORM(s_v, w)
22:         else if ||w, s_w|| < ||v, s_v|| and ||w, s_w|| < ||w, m|| then
23:             insert s_w
24:             CONFORM(v, s_w)
25:         else
26:             insert m
27:         end if
28:     end if
29:     re-embed all conforming edges removed by this call
30: end procedure
```

Related methods generally prove this using the *local feature size* (lfs) [Rup95]. Given a PLC X, *lfs*(p) of any point p in space is the radius of the smallest ball centered on p that intersects two non-intersecting vertices or segments of X. Note that this measure is independent of Steiner points. Termination can then be proven by showing that there is some constant c such that after any Steiner point insertion $c\|e\| \geq \text{lfs}(p)$ holds for any edge e and any point p on e.

However, many of these proofs incorrectly discount the influence of Steiner points. Consider the example of applying Shewchuk [She02] to a problematic case shown in Figure 8. The gray disks show the shortest edge lengths and four times that distance. Recall that for $\text{lfs}(p)$ both Steiner points and intersecting segments of the PLC are ignored. This means that for most points x inside the wedge of conforming segments, $\text{lfs}(x)$ is the distance to p. For this reason, we define the *star local feature size* ($\text{lfs}^\star$) for subspaces.

Definition 1 *Given a fixed PLC X and a subspace S, the star local feature size $\text{lfs}^\star_X(S)$, or simply $\text{lfs}^\star(S)$, of S is the radius of the smallest ball centered within S that contains two points or segments of X that do not intersect in S.*

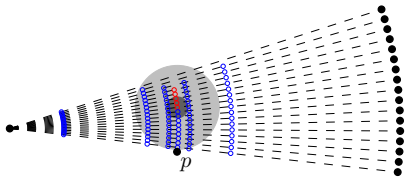

Figure 8: *Shewchuk's edge protection applied to a problem case. The PLC contains the black disks and dashed segments, the small circles show the Steiner points constructed to embed the segments. The red Steiner points are incident to two edges that are too short, the gray disks show the edge length and four times this distance disproving his Theorem 2.*

We are mainly interested in the cases where the subspace is an edge on a segment of X. Note that for edges incident to an acute vertex, $\text{lfs}^\star$ is the minimum lfs over the points on the edge. For other edges, all segments except its supporting segment influence $\text{lfs}^\star$.

Even with this definition, we cannot bound the minimum edge length for our method: case i) can produce arbitrarily short edges when s_v and s_w lie very close to the boundary of the ball. Similarly, in case ii) s_v can lie arbitrarily close to w. Nonetheless, the following three lemmas prove our method always terminates. Their proofs are given in the appendix.

Lemma 2 *Given a PLC X and a segment $\overline{vw}$ of X not embedded in the DT, let us apply the protection method to conform $\overline{vw}$ without re-embedding edges on other segments that were destroyed. This operation can add only a number of Steiner points linear in the number of vertices of the DT.*

Proof Any time a Steiner point is placed on any segment $\overline{vw}$ by the protection method, this creates a segment $\overline{vs}$ based on another point r. As witnessed by the emptiness of the diametral ball touching r, $\overline{vs}$ cannot be destroyed while conforming $\overline{sw}$. It is also straightforward to see that r cannot be touched by any ball with part of $\overline{sw}$ as diameter.
This means that in case i) two points can no longer be inside $\mathcal{D}_{\overline{s_v s_w}}$. In case ii) there is one point that can no longer be in $\mathcal{D}_{\overline{s_v w}}$. In case iii) there are two options: either we insert s_v and one point can no longer be in $\mathcal{D}_{\overline{s_v w}}$, or we insert m. We only insert m if it lies on both $\overline{vs_v}$ and $\overline{ws_w}$, meaning that both $\mathcal{D}_{\overline{vm}}$ and $\mathcal{D}_{\overline{mw}}$. must be empty.
Each operation must either terminate the algorithm, or both construct a segment that will not be destroyed and remove a least one point from the collection of points encroaching upon the remaining part of the segment. This collection starts with a linear number of points and once it is empty the segment must exist in the DT. □

Lemma 3 *Given a PLC X and a collection of Steiner points S constructed using the protection method, there cannot be a point $s \in S$ and an acute vertex v such that $\|s,v\| < \frac{1}{2}lfs(v)$.*

Proof We will prove this by contradiction. Let us assume there is a Steiner point s and an acute vertex v such that $\|s,v\| < \frac{1}{2}\text{lfs}(v)$ and let us consider the different cases in which s could have been constructed.
First let us consider the case where s does not lie on a segment incident to v. By the definition of the local feature size, $\|s,v\| \geq \frac{1}{2}\text{lfs}(v)$.
If s lies on a segment $\overline{vw}$ incident to v, it could only have been constructed when conforming this segment. It could have been constructed to protect either v, w, or any Steiner point on $\overline{vw}$ in any of the three cases. If there is a Steiner point s' between v and s, let us consider this point instead, because $\|v,s'\| < \|v,s\|$.
Recall that according to Lemma 1, from any Steiner point s constructed by our protection method on any segment $\overline{vw}$ from point w and r, $\|v,s\| > \|v,r\|$. This means that s must lie outside $\mathcal{B}_v^r$. Either r is a Steiner point, or r is a point of X and it is straightforward that $\|v,s\| > \|v,r\| \geq \text{lfs}(v)$. If r is a Steiner point, the same reasoning can be applied to find the point r' used to construct r. Because each next point must be closer to v, at some point we will find a non-Steiner point r_i' such that $\|v,s\| > \|v,r_i'\| \geq \text{lfs}(v)$.
This means that in case i) and ii), no Steiner point can be constructed closer to v than $\text{lfs}(v)$. In case iii), either v is not acute, or w is also acute. By our assumption, w must be acute and therefore cannot be a Steiner point. in this case, s must have been constructed such that $\|v,s\| \geq \mathcal{B}_v$ and $\|v,s\| \geq \frac{\|v,w\|}{2}$. In either case, $\|v,s\| \geq \frac{1}{2}\text{lfs}(v)$.
All possible cases result in a distance $\|v,s\| \geq \frac{1}{2}\text{lfs}(v)$. As this contradiction holds for any combination of v and s, our proof is complete. □

Lemma 4 *Given a PLC X and a collection of Steiner points S constructed using the protection method, any edge e embedding a segment of X and not incident to an acute point can only be removed from the DT if $\text{lfs}^\star(e) < \|e\|$.*

Proof This proof follows directly from three facts: 1) e can only be removed if there is a Steiner point s constructed inside the ball with e as diameter, 2) by its construction s must lie on a segment t of X, and 3) t cannot intersect e, because e is not incident to an acute point. □

Theorem 5 *Our protection method always terminates.*

Proof By Lemma 3, edges incident to an acute point have a minimum edge length. By Lemmas 2 and 4 each time an edge is embedded, this adds a finite number of edges and only destroys conforming edges longer than their $\text{lfs}^\star$. □

If we know all segments of the PLC X in advance, we can choose the embedding order such that the occurrence of these short edges is minimized. To achieve this, the Steiner points should be inserted in a strict order. When inserting the next Steiner point s, the segment $\overline{vw}$ and its endpoint v should be chosen such that the length of the conforming edge $\overline{vs}$ resulting from the protection method to v is minimized over all segments and endpoints. In practice, this minimizes the number of changes in $\mathcal{B}_v$ for all point v of X.

3.1.2. Embedding Polygons

After conforming the DT to the segments of the PLC, the interiors of the polygons can be inserted incrementally. Shewchuk [She03] describes a method for recovering the polygons using an ordered sequence of bistellar flips. Unfortunately, this method seems numerically unstable when flips occur almost simultaneously. We incrementally embed the interiors of the polygons using a method that changes larger collections of cells simultaneously very similar to [SG05].

Like Si and Gärtner, we insert each polygon $\mathcal{P}$ per *cavity*. A cavity is a maximal facet-connected collection of cells intersected by $\mathcal{P}$. For each cavity $\mathcal{C}$, we construct two new tetrahedralizations T_a, T_b: one using the vertices of $\mathcal{C}$ on or above $\mathcal{P}$ and the other using the vertices on or below $\mathcal{P}$. When we replace the cells of $\mathcal{C}$ by the overlapping cells of T_a, T_b, the part of $\mathcal{P}$ inside $\mathcal{C}$ appears, because this part can be built from facets on the convex hull of both T_a and T_b.

We must take special caution when embedding polygons near facets constrained earlier. There are simple configurations of points and constraints such that some facets on the boundary of $\mathcal{C}$ are not contained in $T = T_a \cup T_b$. For a relatively simple example of this problem on fifteen points, we refer to [vKvLV13]. When boundary facets are missing from T, the cavity cannot be neatly filled with cells of T.

We overcome this problem similar to Si and Gärtner [SG05]. Before computing T_a, T_b, we grow the cavity until all its boundary facets must be contained in T. We do this by checking for each facet on the boundary of $\mathcal{C}$ whether it has a circumscribing ball that does not contain any point of $\mathcal{C}$. Note that this is a weak Delaunay criterion and unlike [SG05] it is restricted to $\mathcal{C}$. For each facet f not meeting this criterion, we grow $\mathcal{C}$ by adding the cell on the other side of f. We repeat this procedure until all boundary facets meet the criterion. In extreme cases this may force the cavity to grow to the complete CCDT, but in practice most cavities will not grow significantly.

If a cavity $\mathcal{C}$ of a polygon $\mathcal{P}$ to be embedded contains part $\hat{\mathcal{Q}}$ of a polygon embedded earlier, $\hat{\mathcal{Q}}$ should still be embedded after embedding $\mathcal{P}$. In order to retain the embedding of earlier polygons, we embed $\hat{\mathcal{Q}}$ into T_a or T_b, depending on the location of $\hat{\mathcal{Q}}$. The following lemma shows that embedding $\hat{\mathcal{Q}}$ is a strictly smaller problem than embedding $\mathcal{P}$ and so this recursion is finite. The proof is given in the appendix.

Lemma 6 *Given a cavity $\mathcal{C}$ in CCDT T formed while embedding polygon $\mathcal{P}$ and a polygon $\mathcal{Q}$ intersecting $\mathcal{C}$ in $\mathcal{Q}_\mathcal{C}$, let*

T_p be the new CCDT formed by embedding $\mathcal{P}$ without enforcing Q_c to be embedded. The cavity of Q_c in T_c is a strict subspace of C.

Proof We will build this proof in three steps. First, we will show that Q_c is bounded by a loop of coplanar edges of T_c. Then, we will show that the initial cavity of Q_c, before growing it, is a strict subspace of C. Finally, we will show that this cavity can neither grow outside C nor to the other side of $\mathcal{P}$.
In order to show that Q_c is bounded by a loop of coplanar edges of T_c, we will look at the different ways in which Q_c can be bounded. Let us denote the boundary of X by ∂X and recall that Q_c is $Q \cap T_c$. We may divide ∂Q_c into the part ∂_q of ∂Q inside T_c and the part ∂_t of ∂T_c intersection Q. Because Q is embedded in T, ∂_q must be covered by conforming edges and these must also exist in T_c. We may further divide ∂_t into the part ∂_p on $\mathcal{P}$ and the part ∂_c on ∂C. By the PLC criteria, ∂_p must also be covered by conforming edges.
Finally, we show that ∂_c must also be covered by edges in T_c. Let us assume by contradiction that there is a part of ∂_c that is not covered by edges of T_c. This part must then pass through the interior of facets of C. Let us look at an arbitrary facet f of this collection. Note that f intersects Q and f is a facet on the boundary of C. However, recall that ∂C was grown in the cells of T and can only contain facets of T. Because Q is embedded in T, the facets of T cannot transversely intersect Q, a contradiction. Because we took an arbitrary facet f, this contradiction holds for all facets intersected in their interior by ∂_c.
Next, we will show that the initial cavity of Q_c is a strict subspace of C. Recall that the initial cavity is the collection of cells I of T_c intersected by Q_c. As shown above, this intersection is bounded by edges of T_c. In fact, all these edges must also be edges of C. Because Q_c is strictly inside C, I must be a subspace of C. Because Q_c must lie inside T_c and T_c lies on one size of $\mathcal{P}$, I must be a strict subspace of C.
Finally, we will show that the cavity of Q_c can neither grow outside C nor to the other side of $\mathcal{P}$. Recall that we grow the cavity within T_c one cell at a time at facets that are not weakly Delaunay. In order for the cavity to grow outside C, it must at some point have grown through a boundary facet of C, since T_c contains all these boundary facets. However, since all these facets are weakly Delaunay, the cavity cannot grow outside of C. Since we grow the cavity in the cells of T_c and T_c does not contain a point on the other side of $\mathcal{P}$, the cavity cannot grow to the other size of $\mathcal{P}$. □

3.2. Minimum-Weight Graph-Cut

We use a minimum-weight graph-cut algorithm to determine which regions are inside or outside of the objects. Given a graph $G\langle N,E\rangle$ with a set of nodes N containing special source and sink nodes s,s' and a set of directed edges E with weights $w(e) \mapsto \mathcal{R}^+$, let $p(v,\langle X\rangle) \mapsto N$ give the node reached by traversing the directed path of edges $\langle X\rangle$ starting from v. A graph-cut, or simply cut, is a set $C \subseteq E$ such that

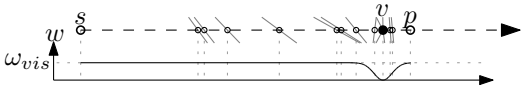

Figure 9: *The influence of visibility on the weights.*

no $p(s,\langle X\rangle) = s'$ exists in $G\langle N, E \setminus C\rangle$. A minimum-weight graph-cut C^- minimizes

$$\sum_{e \in C^-} w(e) \tag{5}$$

We use the method developed by Boykov and Kolmogorov [BK04] to compute the minimum-weight graph-cut, which consistently outperforms the other state-of-the-art methods [Din70, GT88] in practice.

The result of a minimum-weight graph-cut depends on the connectivity of the graph, and its weights. The dual of the CCDT defines the connectivity between all nodes other than the source and sink nodes. The connectivity with the source and sink nodes and the weights of all edges is determined by the geometry of the scene and the measurement process, as described in the next two subsections.

3.2.1. Visibility

From the features of the sensor, we can infer that the line segment connecting data point and sensor can only pass through (semi-)transparent space. We use this information to adjust the weights of the graph the same as Labatut *et al.* [LPK09].

For each data point v and its sensor location s, we estimate a point p inside the object. We place p on the ray $\overrightarrow{sv}$ at a distance 3σ beyond v, where σ is a parameter based upon the measurement precision and the expected minimum thickness of the object. We connect $n(s)$ to the source node and $n(p)$ to the sink node, both with a weight of ω_{vis}, where $n(x)$ is the node in the graph of the CCDT cell containing x.

We know from the scanning procedure that a laser traveling along $\overrightarrow{sp}$ hit a solid surface close to v. We express the exterior space traversed by increasing the weights of the graph. Each edge corresponding to a facet f of the CCDT intersected by $\overline{ps}$ gains a weight $w_{vis}(f) = \omega_{vis}(1 - e^{-d^2/2\sigma^2})$, where $d = \|v,q\|$ and q is the intersection of f and $\overline{ps}$. Note that this weight approaches 0 near v, and ω_{vis} at $d > 3\sigma$, as shown in Figure 9. These weights drive the graph-cut to include a facet near v.

This procedure is repeated for each data point, not only the CCDT vertices, and in all cases the assigned weights are aggregated. This resembles a voting procedure using evidence for the surface location and interior/exterior regions.

3.2.2. Facet Quality

The visibility-based weights are not very robust to degenerate CCDT geometry. For example, long and thin cells are

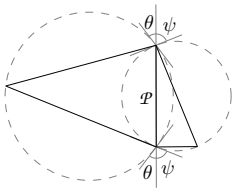

Figure 10: *The angles that determine the facet quality.*

easily missed by all lines-of-sight, meaning they have a large chance of being mislabeled.

We use some inherent quality of each CCDT facet f and its local geometry to adjust its weight. Like Labatut *et al.* [LPK09], we add $w_{qual}(f) = 1 - min(\cos(\theta), \cos(\psi))$ to the weight of each facet, where θ and ψ are the angles between the plane $\mathcal{P}$ supporting f and the circumspheres of the two incident cells at the circle where they intersect $\mathcal{P}$, as shown in Figure 10.

Constrained facets are generally better qualified to be part of the surface. Irrespective of the visibility and facet quality terms, constrained facets are assigned a weight 0. Otherwise we use the aggregated weight $w(f) = w_{qual} + \sum w_{vis}$.

3.3. Implementation

We have implemented our method in C++ using CGAL 4.0.2 [Com13] and an implementation of Boykov and Kolmogorov's graph-cut method [Gra12]. In order to construct a CCDT of a collection of points and polygons, we have built two classes on top of CGAL's Delaunay tetrahedralization: the conforming and the constrained Delaunay tetrahedralization. Both of these classes are straightforward to program based on Section 3.1, although careful programming is required to correctly handle degeneracies, similar to the DT.

4. Experimental Setup

Unfortunately, there are no benchmark data sets available of urban point data, let alone benchmark data with known lines of sight. We use a data set provided by Fugro [Fug12]. This data set contains aerial LiDAR data points that were measured during several flights. Each data point has a registered 3D coordinate and a time stamp that can be matched with the flight path to estimate sensor positions. The massive data sets are separated into smaller chunks based on a regular horizontal grid, with all points collected in one region combined into one chunk. We perform our experiments on seven different chunks, three of which are shown in Figure 11.

Our 32-bit implementation experienced memory problems with excessive data sizes. Therefore, we subsample the data by superimposing a 3D grid with fixed cell dimensions. We sample one random data point per grid cell [ASS13].

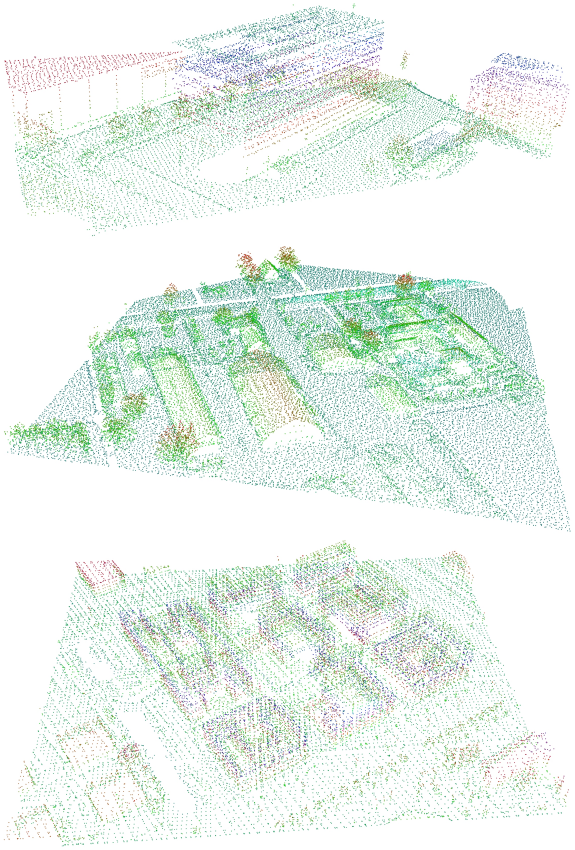

Figure 11: *The points of chunks 1, 3, and 6.*

	$\lvert P_d\rvert$	$\lvert P_s\rvert$	$\lvert S\rvert$		$\lvert P_d\rvert$	$\lvert P_s\rvert$	$\lvert S\rvert$
1	471,203	17,194	42	5	866,767	32,549	51
2	2,199,122	145,201	106	6	6,493,599	28,759	184
3	436,297	42,067	69	7	204,160	20,648	30
4	1,094,277	27,165	25				

Table 1: *The sizes of the chunks: the number of points ($\lvert P_d\rvert$), the number of points after subsampling ($\lvert P_s\rvert$), and the number of shapes ($\lvert S\rvert$) identified by Efficient RANSAC.*

Most chunks use a grid edge length of 1 m, but chunk 6 uses 3 m.

From this subsample, we estimate the local surface normal at each point using PCA on its 12 nearest neighbors. Subsequently, we apply Efficient RANSAC to the chunk to identify planar clusters and the remaining points. Simultaneously, the least-squares optimal approximating plane is computed. We have determined the parameter settings of Efficient RANSAC empirically based on clustering performance. We settled on support threshold 6.5 cm, normal

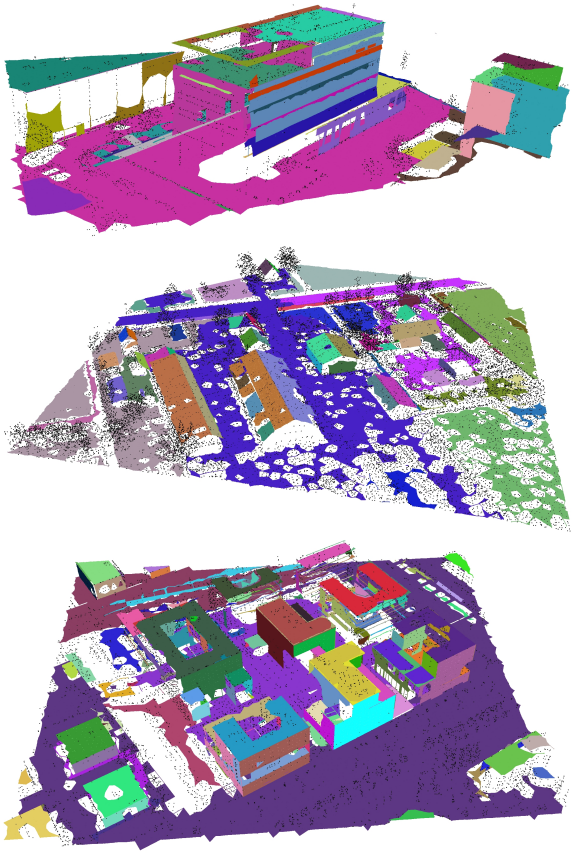

Figure 12: *The guided α-shapes of chunks 1, 3, and 6.*

threshold 20°, bitmap size 1.5 m (2 m on chunk 1 and 4 m on chunk 6), minimal support 25 (10 on chunk 6), and probability 0.0001 of missing a better cluster. Table 1 shows the number of points and shapes per chunks.

For each pair of surfaces with data points within 1 m of each other, the intersection line is computed. We compute the guided α-shape [vKvLV11] of each cluster using the intersection lines of that cluster as guides with the α-value equal to the RANSAC bitmap size. The resulting polygons and the remaining unclustered points are shown in Figure 12.

To compare our method to Labatut *et al.* [LPK09], we construct both the DT and CCDT of the chunk. We insert all the data points into the DT. We insert the collection of guided α-shapes and the unclustered points into the CCDT. Note that for the planar clusters, the CCDT will only contain the points on the boundary. The CCDT also contains a number of Steiner points.

We construct a graph of these arrangements and we determine a graph-cut as explained in Section 3.2. This cut determines the facets that comprise the surfaces of the scene.

	DT				CCDT					
	$\|V\|$	$\|F\|$	$\|C\|$	$\|T\|$	$\|V\|$	$\|S\|$	$\|F\|$	$\|C\|$	$\|T\|$	$\|T\|_r$
1	17k	220k	110k	29k	12k	4k	158k	79k	16k	0.55
2	145k	1874k	937k	197k	109k	26k	1398k	699k	119k	0.61
3	42k	534k	267k	69k	36k	7k	463k	232k	52k	0.75
4	27k	343k	172k	42k	14k	1k	188k	94k	20k	0.46
5	33k	416k	208k	50k	23k	6k	295k	148k	29k	0.59
6	29k	371k	185k	52k	46k	29k	552k	276k	59k	1.12
7	21k	268k	134k	33k	16k	1k	208k	104k	23k	0.70

Table 2: *The number of vertices ($|V|$), of which Steiner ($|S|$), facets ($|F|$), and cells ($|C|$) in the DT and CCDT for the various chunks, and the number of triangles of the watertight surface ($|T|$). Each of these values is given in thousands. Finally, the ratio of triangles between CCDT and DT ($|T|_r$).*

5. Results

Figure 11 shows some of the chunks of urban LiDAR we have performed our experiments on. Figure 13 shows the watertight geometry constructed using [LPK09] and our method. It is clear that our method constructs geometry that is simpler and that conforms more to our idea what an urban scene should look like. The small details are represented by free-form meshes, while the large roofs and facades are represented by flat surfaces. An unfortunate side-effect to the concise description of the planar surfaces is that it makes the other, noisy regions stand out more.

Note that vegetation is also reconstructed in various places. The quality of the reconstruction of vegetation is based both on the number of points in the trees and bushes and on the value of σ. Larger values of σ increase the chance that cells outside the object are connected to the sink. This results in this connection being cut, instead of an edge between Delaunay cells.

Apart from producing a geometric model that better captures the planar nature of an urban scene, this geometry is also generally more concise, as Table 2 shows. In most cases, the CCDT has roughly a sixth fewer elements and the watertight surface generated from it uses roughly a third fewer triangles than when using the DT.

Chunk 6 is an exception in this regard: in this case the CCDT-based surface uses more triangles. We expect this is caused by the much higher than average shape to point ratio, as shown in Table 1. This chunk has roughly three times as many shapes as chunks of similar size. It is not surprising that the reconstruction would require more triangles to cover these shapes.

The disadvantage of our method is that it takes significantly more time to compute the geometry, as shown in Table 3. Once the CCDT is computed, traversing the lines of sight to set the weights of the graph and performing the graph-cut takes roughly the same amount of time for the CCDT as for the DT. We are currently working on improving the efficiency of constructing the CCDT.

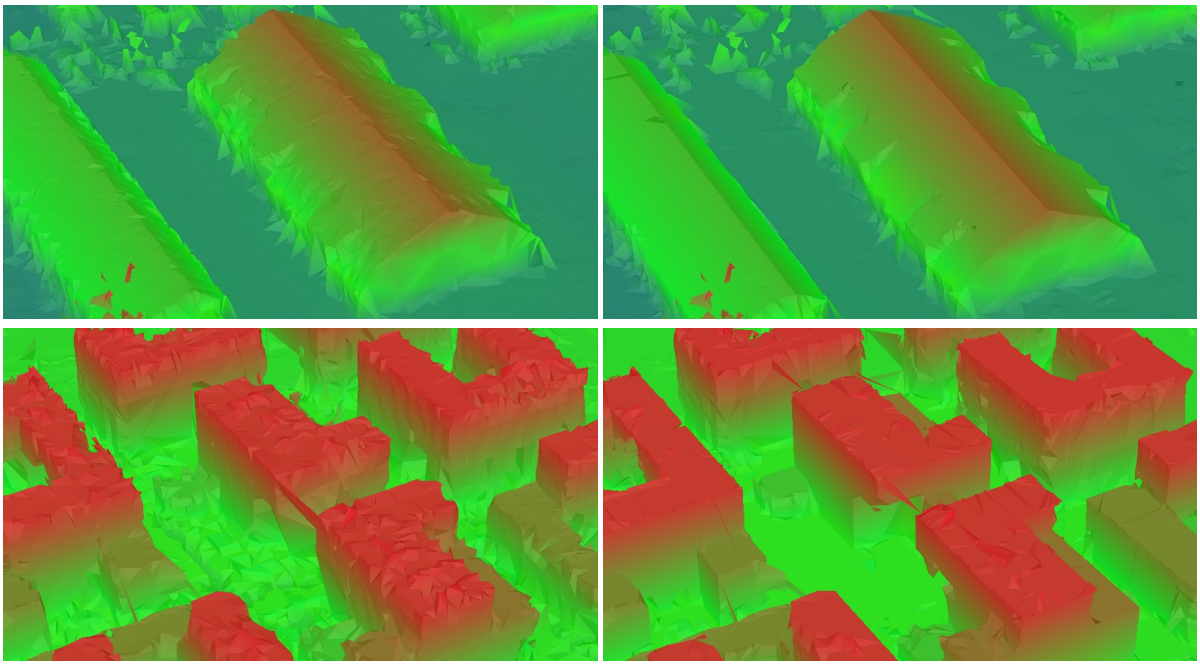

Figure 13: *Details of the watertight geometry produced using [LPK09] (left), and our method (right) of chunks 3 (top) and 6 (bottom); the results on chunk 1 are shown in Figure 1. The colors are based on vertex height and triangle normal.*

	DT				CCDT						
	T	t_P	t_L	t_G	T	t_P	t_l	t_b	t_p	t_L	t_G
1	5.11	12%	64%	25%	37.68	24%	13%	21%	34%	7%	2%
2	48.20	10%	64%	26%	442.56	30%	36%	11%	15%	6%	2%
3	12.16	10%	64%	25%	87.78	45%	12%	12%	20%	8%	3%
4	7.92	11%	64%	25%	20.84	30%	5%	16%	28%	16%	5%
5	9.27	11%	64%	26%	54.27	25%	25%	19%	19%	8%	3%
6	8.13	13%	61%	27%	361.86	34%	29%	21%	14%	1%	1%
7	5.63	12%	62%	27%	17.48	45%	10%	10%	13%	16%	6%

Table 3: *The total time (T) in seconds to compute the watertight geometry using the DT and CCDT and the percentage of time taken by the steps in the process: inserting the points (t_P), inserting the intersection lines between shapes (t_l), inserting the boundaries of the shapes (t_b), inserting the interiors of the shapes (t_p), traversing the lines of sight to adjust the weights (t_L), and performing the graph-cut (t_G).*

6. Conclusions

We have presented a method to construct watertight geometry for urban scenes. Our method extends an earlier method by incorporating the assumption that many of the surfaces in urban scenes are piecewise planar. The method achieves this by performing a minimum-weight graph-cut on a conforming constrained Delaunay tetrahedralization. Although our method is slower than the earlier method that uses the Delaunay tetrahedralization, the results are more concise and they are more visually pleasing.

One of the most interesting directions to improve on this method is to reduce the time needed to compute the CCDT, as this is an order of magnitude worse than computing the DT. Some clever usage of multi-core processing may greatly speed up the process.

Our method reconstructs the scene as a whole, ignoring the properties of different objects in the scene. Tuning the parameters to the different objects may produce better results. For example, we may wish to reduce σ for points on vegetation to indicate the smaller expected object width.

Finally, a major issue in urban reconstruction is the general lack of ground truths and benchmark data sets. This makes it difficult to quantify results. We have expressed the quality of our method in the number of triangles. However, establishing benchmarks and comparing various methods on these will increase insight into their respective strengths.

Acknowledgments This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie). We would like to thank Pierre Alliez, Ramsay Dyer, and Mariette Yvinec for their insightful comments.

References

[AAK*09] AICHHOLZER O., AURENHAMMER F., KORNBERGER B., PLANTINGA S., ROTE G., STURM A., VEGTER

G.: Recovering structure from r-sampled objects. *Comp. Graph. Forum 28*, 5 (2009), 1349–1360. 2

[ACDL00] AMENTA N., CHOI S., DEY T. K., LEEKHA N.: A simple algorithm for homeomorphic surface reconstruction. In *SoCG* (2000), pp. 213–222. 2

[ACSTD07] ALLIEZ P., COHEN-STEINER D., TONG Y., DESBRUN M.: Voronoi-based variational reconstruction of unoriented point sets. In *SGP* (2007), pp. 39–48. 2

[ASS13] ALLIEZ P., SABORET L., SALMAN N.: Point set processing. 2013. http://www.cgal.org/Manual/4.0/doc_html/cgal_manual/Point_set_processing_3/Chapter_main.html. 9

[BK04] BOYKOV Y., KOLMOGOROV V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI 26*, 9 (2004), 1124–1137. 8

[Boi84] BOISSONNAT J.-D.: Geometric structures for three-dimensional shape representation. *ACM Trans. Graph. 3*, 4 (1984), 266–286. 2

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3D objects with radial basis functions. In *SIGGRAPH* (2001), pp. 67–76. 2

[CG06] CAZALS F., GIESEN J.: Delaunay triangulation based surface reconstruction. In *Effective Computational Geometry for Curves and Surfaces*. 2006, pp. 231–276. 2

[CLP10] CHAUVE A.-L., LABATUT P., PONS J.-P.: Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data. In *CVPR* (2010), pp. 1261–1268. 2

[Com13] COMPUTATIONAL GEOMETRY ALGORITHMS LIBRARY:. http://www.cgal.org/, 2013. 9

[DG03] DEY T. K., GOSWAMI S.: Tight cocone: a water-tight surface reconstructor. In *SM '03* (2003), pp. 127–134. 2

[Din70] DINIC E. A.: Algorithm for solution of a problem of maximum flow in networks with power estimation. *Soviet Mathematics Doklady 11* (1970), 1277–1280. 8

[EKS83] EDELSBRUNNER H., KIRKPATRICK D. G., SEIDEL R.: On the shape of a set of points in the plane. In *IEEE Trans. Inf. Th.* (1983), vol. 29, pp. 551–559. 2

[Fug12] FUGRO EARTHDATA, INC.:. http://www.fugroearthdata.com/, 2012. 9

[Gra12] GRAPH CUT OPTIMIZATION LIBRARY:. http://cbia.fi.muni.cz/user_dirs/gc_doc/, 2012. 9

[GT88] GOLDBERG A. V., TARJAN R. E.: A new approach to the maximum-flow problem. *J. ACM 35*, 4 (1988), 921–940. 8

[LPK09] LABATUT P., PONS J.-P., KERIVEN R.: Robust and efficient surface reconstruction from range data. *Comp. Graph. Forum 28*, 8 (2009), 2275–2290. 2, 8, 9, 10, 11

[MDGD*10] MULLEN P., DE GOES F., DESBRUN M., COHEN-STEINER D., ALLIEZ P.: Signing the unsigned: robust surface reconstruction from raw pointsets. *Comp. Graph. Forum 29*, 5 (2010), 1733–1741. 2

[MTT*96] MILLER G. L., TALMOR D., TENG S.-H., WALKINGTON N., WANG H.: Control volume meshes using sphere packing: generation, refinement and coarsening. In *IMR* (1996), pp. 47–61. 3

[Rup95] RUPPERT J.: A delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Alg. 18*, 3 (1995), 548–585. 6

[SDK09] SCHNABEL R., DEGENER P., KLEIN R.: Completion and reconstruction with primitive shapes. *Comp. Graph. Forum 28* (2009), 503–512. 2

[SG05] SI H., GÄRTNER K.: Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. In *IMR* (2005), pp. 147–163. 2, 3, 5, 7

[She98] SHEWCHUK J. R.: A condition guaranteeing the existence of higher-dimensional constrained delaunay triangulations. In *SoCG* (1998), pp. 76–85. 2, 3

[She02] SHEWCHUK J. R.: Constrained Delaunay tetrahedralizations and provably good boundary recovery. In *IMR* (2002), pp. 193–204. 2, 3, 5, 6

[She03] SHEWCHUK J. R.: Updating and constructing constrained Delaunay and constrained regular triangulations by flips. In *SoCG* (2003), pp. 181–190. 7

[SOS04] SHEN C., O'BRIEN J. F., SHEWCHUK J. R.: Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph. 23*, 3 (2004), 896–904. 2

[SSZCO10] SHALOM S., SHAMIR A., ZHANG H., COHEN-OR D.: Cone carving for surface reconstruction. *ACM Trans. Graph. 29*, 6 (2010), 150:1–150:10. 2

[SWK07] SCHNABEL R., WAHL R., KLEIN R.: Efficient RANSAC for point-cloud shape detection. *Comp. Graph. Forum 26*, 2 (2007), 214–226. 2

[TkG07] TARSHA-KURDI F., GRUSSENMEYER P.: Hough-transform and extended RANSAC algorithms for automatic detection of 3D building roof planes from lidar data. In *IAPRS* (2007), vol. 36. 2

[TTC07] TSENG Y.-H., TANG K.-P., CHOU F.-C.: Surface reconstruction from LiDAR data with extended snake theory. In *EMMCVPR '07* (2007), pp. 479–492. 2

[vKvLV11] VAN KREVELD M., VAN LANKVELD T., VELTKAMP R. C.: On the shape of a set of points and lines in the plane. *Comp. Graph. Forum 30*, 5 (2011), 1553–1562. 2, 10

[vKvLV13] VAN KREVELD M., VAN LANKVELD T., VELTKAMP R. C.: *Watertight Scenes from Urban LiDAR and Planar Surfaces*. Tech. Rep. UU-CS-2013-007, Utrecht University, Department of Information and Computing Sciences, 2013. 7